

Projeto de Implementação – Agentes Inteligentes e Busca

Disciplina: Inteligência Artificial

Referências obrigatórias:

- Russell & Norvig – *Artificial Intelligence: A Modern Approach* (AIMA)
- Repositório oficial: <https://github.com/aimacode/aima-python>

SUBMISSÃO: SOMENTE um representante do grupo submete os entregues via google classroom

1. Objetivo do Projeto

O objetivo deste projeto é implementar um **agente inteligente baseado em busca que solucione um problema proposto pelo grupo**, utilizando os conceitos teóricos estudados na disciplina (arquitetura Ambiente – Agente – Programa de Agente) e as classes base do repositório **aima-python**, conforme apresentado no livro *Artificial Intelligence: A Modern Approach* (AIMA).

Cada grupo deverá **propor, modelar e implementar um problema de escolha livre**, desde que:

- o problema seja **original** (não pode ser um problema já resolvido no livro AIMA nem no repositório aima-python);
- a solução utilize explicitamente a arquitetura **Ambiente – Agente – Programa de Agente**;
- os **algoritmos de busca** sejam utilizados **dentro do programa do agente**, e não apenas como chamadas isoladas.

O grupo é responsável por **justificar todas as decisões de modelagem**, escolhas de algoritmos e limitações da solução proposta.

2. Especificação Formal do Problema (Obrigatória)

Cada grupo deve apresentar a **especificação formal do problema**, conforme o modelo clássico do AIMA, contendo explicitamente:

- Representação dos **estados**
- **Estado inicial**
- Conjunto de **ações**
- **Modelo de transição** (`result(s, a)`)
- **Teste de objetivo** (`goal_test`)
- **Custo de caminho** (`path_cost`)

♦ **Obrigatório:**

Cada item da especificação deve ser **claramente mapeado para o código correspondente**, indicando onde e como foi implementado.

3. Classificação do Ambiente (Obrigatória)

O grupo deve classificar formalmente o ambiente do problema proposto, justificando cada item, segundo os critérios do AIMA:

- Determinístico ou estocástico
 - Totalmente ou parcialmente observável
 - Estático ou dinâmico
 - Discreto ou contínuo
 - Agente único ou múltiplos agentes
-

4. Arquitetura Ambiente – Agente – Programa de Agente

A implementação deve respeitar explicitamente a separação conceitual:

- **Ambiente**: responsável por manter o estado do mundo, fornecer percepções e executar ações.
- **Agente**: entidade inserida no ambiente.
- **Programa de Agente**: função ou classe responsável por decidir a ação a partir da percepção.

♦ **Atenção (conceito central do projeto):**

O **programa de agente não é o algoritmo de busca em si** (embora ele seja necessário). Ele deve:

- receber percepções do ambiente;
- decidir quando formular um problema;
- executar um algoritmo de busca para gerar um **plano (sequência de ações)**;
- retornar **uma ação por passo** ao ambiente.

Os algoritmos de busca (BFS, A*, etc.) devem ser utilizados **dentro do programa de agente**, conforme discutido em aula e conforme a classe `SimpleProblemSolvingAgentProgram` do `aima-python`.

5. Uso do Repositório `aima-python`

- Devem ser utilizadas as **classes base do `aima-python`**, criando subclasses sempre que necessário (por exemplo: `Problem`, `Environment`, programas de agente).
 - Não é permitido “reescrever” do zero estruturas que já existem no repositório sem justificativa.
-

6. Algoritmos de Busca e Heurísticas

- Devem ser utilizados os **algoritmos de busca vistos em sala de aula** que existam no repositório `aima-python`.
- Na apresentação final, o grupo deve:
 - listar **quais algoritmos foram utilizados**;
 - listar **quais algoritmos não foram utilizados**;
 - **justificar** por que determinados algoritmos não são adequados ao problema proposto.

Heurísticas

- Para algoritmos informados, o grupo deve:
 - definir explicitamente a heurística $h(n)$;
 - explicar sua intuição;
 - discutir se ela é admissível e/ou consistente (mesmo que informalmente);
 - analisar seu impacto no desempenho do agente.
-

7. Testes e Visualização

- Devem ser criados **testes automatizados** (por exemplo, com `pytest`) para verificar o funcionamento da implementação.
- O ambiente deve implementar um método `render()`, que imprima o estado do ambiente a cada passo, conforme visto nos exemplos em aula.

8. Estrutura Mínima do Projeto

O projeto deve seguir, no mínimo, a seguinte organização:

```
/projectGrupo1
  /env      (ambiente)
  /agents    (programa(s) do agente)
  /problems (subclasses de Problem)
  /tests
  main.py
  README.md
```

O `README.md` deve conter instruções claras para execução do projeto.

9. Apresentação e Entrega

- A avaliação será **presencial** (somente integrantes dos grupos presentes serão avaliados).
- A entrega consiste em:
 - código-fonte ou repositório;
 - **vídeo curto (3–5 minutos)** demonstrando o agente em execução no ambiente e explicando a solução.
- O grupo deve garantir que a solução **rode durante a avaliação** (gerenciar notebook próprio ou execução prévia no laboratório).

Slide obrigatório

- O colocar **UM slide da apresentação (na apresentação anterior de cada grupo)** deve conter:
 - título do projeto;
 - breve descrição do problema proposto.

Esse slide será usado para evitar que diferentes grupos implementem o mesmo problema.

10. Critérios de Avaliação

A avaliação considerará:

- Modelagem correta do problema (estados, ações, objetivo)
 - Arquitetura ambiente–agente–programa de agente
 - Uso correto e justificado dos algoritmos de busca
 - Implementação e análise de heurísticas
 - Qualidade dos testes automatizados
 - Clareza da apresentação e do vídeo
 - Capacidade do grupo de explicar e executar o sistema presencialmente
-

Observação Final

O problema é de escolha livre, mas **não é permitido reutilizar problemas já resolvidos no livro AIMA ou no repositório aima-python.**

O foco do projeto está na **integração entre teoria e implementação**, especialmente na correta utilização de **busca como mecanismo de decisão dentro de um programa de agente**.